UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/536,745	05/27/2005	Gregory R Bentz	13210-209	4513

1059 7590 11/21/2008

BERESKIN AND PARR
40 KING STREET WEST
BOX 401
TORONTO, ON M5H 3Y2
CANADA

EXAMINER	
JORDAN, KIMBERLY L	
ART UNIT	PAPER NUMBER
4114	

MAIL DATE	DELIVERY MODE
11/21/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/536,745	BENTZ ET AL.	
	Examiner	Art Unit	
	Kimberly Jordan	4114	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *27 May 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-18* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-18* is/are rejected.

7)☒ Claim(s) *2,7,8, and 14* is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *5/15/08; 11/23/07; 5/17/07; 5/17/07; 11/24/06; 4/12/06; 11/01/05*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This is the initial Office action based on the application filed on May 27, 2005.

2. Claims 1-18 are pending.

Information Disclosure Statement

3. The Information Disclosure Statements filed *5/15/08; 11/23/07; 5/17/07; 5/17/07; 11/24/06; 4/12/06; 11/01/0* have been considered. Initialed copies of the Form 1449 are enclosed herewith.

Claim Objections

4. **Claim 2** is objected to because of the following informalities:

- **Claim 2** recites the limitation "the cod file". The Examiner subsequently interprets this limitation as reading "the given cod file" for the purpose of providing it with proper explicit antecedent basis.

Appropriate correction is required.

5. **Claim 7** is objected to because of the following informalities:

- **Claim 7** recites the limitation "constant pool entries from the class files". The Examiner subsequently interprets this limitation as reading "constant pool entries from the plurality of class files" for the purpose of providing it with proper explicit antecedent basis.
- **Claim 7** recites the limitation "information structure entries from the class files". The Examiner subsequently interprets this limitation as reading "information structure entries from the plurality of class files" for the purpose of providing it with proper explicit antecedent basis.

Appropriate correction is required.

6. **Claim 8** is objected to because of the following informalities:

- **Claim 8** recites the limitation "the cod file". The Examiner subsequently interprets this limitation as reading "the given cod file" for the purpose of providing it with proper explicit antecedent basis.

Appropriate correction is required.

7. **Claim 14** is objected to because of the following informalities:

- **Claim 14** recites the limitation "the cod file". The Examiner subsequently interprets this limitation as reading "the given cod file" for the purpose of providing it with proper explicit antecedent basis.

Appropriate correction is required.

Claim Rejections - 35 USC § 112

8. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

9. Claims 1 and 7-12 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 1 appears to be reciting intended use in a method step within a device claim. The structure of the device is therefore not clear. The Examiner subsequently interprets this claim as reading " A device comprising:

- a memory unit including executable software; a plurality of class files stored in the memory unit; and,
- a computing unit connected to the memory unit, the computing unit being able to execute a Java Virtual Machine, where a given cod file generated from a class file includes:

- a constant pool created by combining constant pool entries from two or more of the class files without duplication of entries;
- a byte codes and information structure created by combining byte codes and information structure entries from the two or more of the class files without duplication of entries; and,
- a fixup table for providing information to the Java Virtual Machine for resolving at least one entry in the given cod file at link time."

10. Claim 7 recites the limitation "in at least one of the constant pool and the byte codes and information structure". There is insufficient antecedent basis for this limitation in the claim.

11. Claims 8-12 depend on Claim 7 and, therefore, suffer the same deficiency as Claim 7.

Claim Rejections - 35 USC § 101

12. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

13. **Claims 7-18** are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

14. **Claims 7-12** are directed to a method. However, the recited steps of the method are held to be non-statutory subject matter because the recited steps of the method are (1) not tied to another statutory class (such as a particular apparatus) or (2) not transforming the underlying subject matter (such as an article or materials) to a different state or thing.

15. **Claims 13-18** are directed to an article. However, the article appears to lack the necessary physical components (hardware) to constitute a machine or manufacture under § 101. An article could reasonably be construed to mean carrier wave and is thus an intangible medium. Therefore, these claim limitations can be reasonably interpreted as computer program modules or software *per se*. The claim is directed to functional descriptive material *per se* and hence non-statutory.

These claims constitute computer programs representing computer listings *per se*. Such descriptions or expressions of the programs are not physical "things". They are neither computer components nor statutory processes, as they are not "acts" being performed. Such claimed computer programs do not define any structural and functional interrelationships between the computer program and other claimed elements of a computer, which permit the computer program's functionality to be realized. In contrast, a claimed computer program embodied on a tangible computer-readable medium is a computer element, which defines structural and functional interrelationships between the computer program and the rest of the computer, that permits the computer program's functionality to be realized, and is thus statutory. See *Lowry*, 32 F.3d at 1583-84, 32 USPQ2d at 1035.

Claim Rejections - 35 USC § 103

16. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

17. **Claims 1-18** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Baentsch et al. (WO 99/49392)**, hereinafter Baentsch, in view of **Swetland (US 2002/0170047 A1).**

As per **Claim 1**, Baentsch discloses:

- *a fixup table for providing information to the Java Virtual Machine for resolving at least one entry in the given cod file at link time* (see at least Figures 1-3; Page 4, lines 8-9, "The cap file also maintains the necessary relocation information in fixup tables."; Page 4, lines 12-13, "The fixup table again contains the position in the text or data section where a relocation has to take place.").

Baentsch does not disclose:

- *a memory unit including executable software;*
- *a plurality of class files stored in the memory unit;*
- *a computing unit connected to the memory unit, the computing unit being able to execute a Java Virtual Machine;*
- *a constant pool created by combining constant pool entries from two or more of the class files without duplication of entries;* and
- *a byte codes and information structure created by combining byte codes and information structure entries from the two or more of the class files without duplication of entries.*

Swetland discloses:

- *a memory unit including executable software* (see at least Figure 7, reference number 750; Paragraph 0046, "The external memory may be used to store programs…")*;*
- *a plurality of class files stored in the memory unit* (see at least Paragraph 0049, "…microprograms and portal data are transmitted from the portal server to the external memory of the portal device.."; "The microprograms in one embodiment are comprised of compact, interpreted instructions known as 'bytecodes,'…"; Paragraph 0077, "As described above, in one embodiment, the bytecodes may be Java bytecodes/applets."; Paragraph 0083, "…each of the class files used in a particular application program…are combined to form a unified programming object referred to herein as a 'bundle'. For the purpose of illustration, the particular bundle...is constructed from the class files.")*;*
- *a computing unit connected to the memory unit* (see at least Figure 7, reference number 710; Paragraph 0047, "The microcontroller of one embodiment is comprised of a central processing unit ("CPU"), a read only memory ("ROM"), and a scratchpad RAM. The ROM is further comprised of an interpreter module and a toolbox module");
- *the computing unit being able to execute a Java Virtual Machine* (see at least Paragraph 0075, "…the interpreter module on the portal device is a Java virtual machine.")*;*

- *a constant pool created by combining constant pool entries from two or more of the class files without duplication of entries* (see at least Paragraph 0083, "…each of the class files used in a particular application program…are combined to form a unified programming object referred to herein as a 'bundle'. For the purpose of illustration, the particular bundle...is constructed from the class files. More specifically, the redundant [method] entries are combined into a single, 'global' [method] entry in a shared constant pool within the bundle." By combining the redundant entries into a unified method, there is no duplication of entries.)*;* and
- *a byte codes and information structure created by combining byte codes and information structure entries from the two or more of the class files without duplication of entries* (see at least Figure 2; Paragraph 0084, "The methods and fields from the original class fields are copied to the bundle as well along with various other class file objects (not shown)." Applicant regards "byte codes and information structure" as the "class properties, the methods, fields and attributes of the class, and their types" (Paragraph 0013). The constant pool contains references to these byte codes and information structures. Therefore, if the constant pool entities have been combined into the bundle without duplication, the methods, fields, and attributes are copied into the bundle without duplication as well.).

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention was made to incorporate Swetland's, memory and computing units and process of merging the constant pools and class structures of related methods, into Baentsch's, fixup table. The modification would be obvious because one of ordinary skill in the art would be motivated to "[reduce] the memory requirements for object-oriented programs" (Swetland, Paragraph 0011), "considering the fact that a program may utilize scores of class files" (Swetland, Paragraph 0010). By merging the constant pools and class structures of related methods, the memory requirements for programs will be reduced since there can be an enormous number of class files, and hence methods, in a program. Baentsch's fixup table allows for "runtime efficient linking in resource

constraint Java runtime environments" (Baentsch, Page 1: 26-27). Applicant's cod file is a compressed class file, and Baentsch's cap file is a package file that contains classes and interfaces. Both files contain class information and for the purposes of combining class information and one of ordinary skill in the art would view these as comparable file types.

As per **Claim 2**, the rejection of **Claim 1** is incorporated; and Baentsch further discloses:

- *wherein for the given cod file, the fixup table includes a symbolic reference for cross-referencing a method not contained in the cod file* (see at least Page 4, lines 22-24, "…references to other external packages should not be linked by precalculated offsets. Instead, a name or identifier should be used for references to other packages during the link process.").

As per **Claim 3**, the rejection of **Claim 1** is incorporated; and Baentsch further discloses:

- *and the fixup table of the given cod file further includes indices to the other related cod files specified in the sibling group* (see at least Page 4, lines 12-14, "The fixup table...contains the position in the text or data section where a relocation has to take place. In the simple case, these places are also relocated by a precalculated offset into trusted and well known target packages.")

Baentsch does not disclose:

- *wherein the given cod file further includes a sibling list for listing other related cod files to define a sibling group*

Swetland discloses:

- *wherein the given cod file further includes a sibling list for listing other related cod files to define a sibling group* (see at least Figure 12; Paragraph 0083, "…each of the class files used in a particular application program…are combined to form a unified programming object referred to herein as a 'bundle'. For the purpose of illustration, the particular

bundle...is constructed from the class files." Related class files are combined to form a bundle, which one of ordinary skill in the art would view as a sibling group.).

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention was made to incorporate Swetland's bundled class files into Baentsch's fixup table containing references to the related files. The modification would be obvious because one of ordinary skill in the art would be motivated to group related class files to form a unified class structure in a medium that limits the size of an individual file by having multiple smaller related files as opposed to a single larger combined file.

As per **Claim 4**, the rejection of **Claim 1** is incorporated; and Baentsch further discloses:

- *wherein for the given cod file, the byte codes and information structure includes a hard offset for cross-referencing a method included in the given cod file that was previously symbolically referenced* (see at least Page 4, lines 12-14, "The fixup table...contains the position in the text or data section where a relocation has to take place. In the simple case, these places are also relocated by a precalculated offset into trusted and well known target packages.").

As per **Claim 5**, the rejection of **Claim 3** is incorporated; and Baentsch further discloses:

- *wherein for the given cod file, the byte codes and information structure includes a hard offset for cross-referencing a method included in the sibling group that was previously symbolically referenced* (see at least Page 4, lines 12-14, "The fixup table...contains the position in the text or data section where a relocation has to take place. In the simple case, these places are also relocated by a precalculated offset into trusted and well known target packages." It would have been obvious to one of ordinary skill in the art to be able to use hard-offsets in a file to reference other related files, or sibling files.).

As per **Claim 6**, the rejection of **Claim 3** is incorporated; and Baentsch further discloses:

- *wherein for the given cod file, the fixup table includes a symbolic reference for cross-referencing a method not contained in the sibling group* (see at least Page 4, lines 22-24, "…references to other external packages should not be linked by precalculated offsets. Instead, a name or identifier should be used for references to other packages during the link process." These references made to external packages can include methods.)

As per **Claim 7**, Baentsch discloses:

- *generating a fixup table for providing information to a Java Virtual Machine for resolving at least one entry in the given cod file at link time* (see at least Figures 1-3; Page 4, lines 8-9, "The cap file also maintains the necessary relocation information in fixup tables."; Page 4, lines 12-13, "The fixup table again contains the position in the text or data section where a relocation has to take place.").

Baentsch does not disclose:

- *identifying class files with common entries in at least one of the constant pool and the byte codes and information structure;*
- *generating a constant pool for the given cod file by combining constant pool entries from the class files with common entries without duplication;* and
- *generating the byte codes and information structure for the given cod file by combining byte codes and information structure entries from the class files with common entries without duplication.*

Swetland discloses:

- *identifying class files with common entries in at least one of the constant pool and the byte codes and information structure* (see at least Paragraph 0083, "…each of the class files used in a particular application program…are combined to form a unified programming object referred to herein as a 'bundle'. …the redundant [method] entries

are combined into a single, 'global' [method] entry in a shared constant pool within the bundle.");

- *generating a constant pool for the given cod file by combining constant pool entries from the class files with common entries without duplication* (see at least Paragraph 0083, "…each of the class files used in a particular application program…are combined to form a unified programming object referred to herein as a 'bundle'. For the purpose of illustration, the particular bundle...is constructed from the class files. More specifically, the redundant [method] entries are combined into a single, 'global' [method] entry in a shared constant pool within the bundle." By combining the redundant entries into a unified method, there is no duplication of entries.); and
- *generating the byte codes and information structure for the given cod file by combining byte codes and information structure entries from the class files with common entries without duplication* (see at least Paragraph 0084, "The methods and fields from the original class fields are copied to the bundle as well along with various other class file objects (not shown)." Applicant regards "byte codes and information structure" as the "class properties, the methods, fields and attributes of the class, and their types" (Paragraph 0013). The constant pool contains references to these byte codes and information structures. Therefore, if the constant pool entities have been combined into the bundle without duplication, the methods, fields, and attributes are copied into the bundle without duplication as well.).

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention was made to incorporate Swetland's, memory and computing units and process of merging the constant pools and class structures of related methods, into Baentsch's, fixup table. The modification would be obvious because one of ordinary skill in the art would be motivated to "[reduce] the memory requirements for object-oriented programs" (Swetland, Paragraph 0011), "considering the fact that a program may utilize scores of class files" (Swetland, Paragraph 0010). By merging the constant pools and class structures of related methods, the memory requirements

for programs will be reduced since there can be an enormous number of class files, and hence methods, in a program. Baentsch's fixup table allows for "runtime efficient linking in resource constraint Java runtime environments" (Baentsch, Page 1: 26-27). Applicant's cod file is essentially a compressed class file, and Baentsch's cap file is a package file that contains classes and interfaces. Both files contain class information and for the purposes of combining class information and one of ordinary skill in the art would view these as comparable file types.

As per **Claim 8**, the rejection of **Claim 7** is incorporated; and Baentsch further discloses:

- *for the given cod file, providing a symbolic reference in the fixup table for cross-referencing a method not contained in the cod file* (see at least Page 4, lines 22-24, "…references to other external packages should not be linked by precalculated offsets. Instead, a name or identifier should be used for references to other packages during the link process.").

As per **Claim 9**, the rejection of **Claim 7** is incorporated; and Baentsch further discloses:

- *providing indices to the other related cod files specified in the sibling group in the fixup table* (see at least Page 4, lines 12-14, "The fixup table...contains the position in the text or data section where a relocation has to take place. In the simple case, these places are also relocated by a precalculated offset into trusted and well known target packages.").

Baentsch does not disclose:

- *generating a sibling list for listing other related cod files to define a sibling group;*

Swetland discloses:

- *generating a sibling list for listing other related cod files to define a sibling group* (see at least Figure 12; Paragraph 0083, "…each of the class files used in a particular application program…are combined to form a unified programming object referred to herein as a 'bundle'. For the purpose of illustration, the particular bundle...is constructed from the

class files." Related class files are combined to form a bundle, which one of ordinary skill in the art would view as comparable to a sibling group.).

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention was made to incorporate Swetland's bundled class files into Baentsch's fixup table containing references to the related files. The modification would be obvious because one of ordinary skill in the art would be motivated to group related class files to form a unified class structure in a medium that limits the size of an individual file by having multiple smaller related files as opposed to a single larger combined file.

As per **Claim 10**, the rejection of **Claim 7** is incorporated; and Baentsch further discloses:

- *for the given cod file, providing a hard offset in the byte codes and information structure for cross-referencing a method included in the given cod file that was previously symbolically referenced* (see at least Page 4, lines 12-14, "The fixup table...contains the position in the text or data section where a relocation has to take place. In the simple case, these places are also relocated by a precalculated offset into trusted and well known target packages.").

As per **Claim 11**, the rejection of **Claim 9** is incorporated; and Baentsch further discloses:

- *for the given cod file, providing a hard offset in the byte codes and information structure for cross-referencing a method included in the sibling group that was previously symbolically referenced* (see at least Page 4, lines 12-14, "The fixup table...contains the position in the text or data section where a relocation has to take place. In the simple case, these places are also relocated by a precalculated offset into trusted and well known target packages." It would have been obvious to one of ordinary skill in the art to be able to use hard-offsets in a file to reference other related files, or sibling files.).

As per **Claim 12**, the rejection of **Claim 9** is incorporated; and Baentsch further discloses:

- *for the given cod file, providing a symbolic reference for the fixup table for cross-referencing a method not contained in the sibling group* (see at least Page 4, lines 22-24, "…references to other external packages should not be linked by precalculated offsets. Instead, a name or identifier should be used for references to other packages during the link process." These references made to external packages can include methods.).

As per **Claim 13**, Baentsch discloses:

- *a fixup table for providing information to a Java Virtual Machine for resolving at least one component of the given cod file at link time* (see at least Figures 1-3; Page 4, lines 8-9, "The cap file also maintains the necessary relocation information in fixup tables."; Page 4, lines 12-13, "The fixup table again contains the position in the text or data section where a relocation has to take place.").

Baentsch does not disclose:

- *a constant pool created by combining constant pool entries from two or more of the class files without duplication of entries;* and
- *a byte codes and information structure created by combining byte codes and information structure entries from the two or more of the class files without duplication of entries.*

Swetland discloses:

- *a constant pool created by combining constant pool entries from two or more of the class files without duplication of entries* (see at least Paragraph 0083, "…each of the class files used in a particular application program…are combined to form a unified programming object referred to herein as a 'bundle'. For the purpose of illustration, the particular bundle...is constructed from the class files. More specifically, the redundant [method] entries are combined into a single, 'global' [method] entry in a shared constant pool within the bundle." By combining the redundant entries into a unified method, there is no duplication of entries.)*;* and

- *a byte codes and information structure created by combining byte codes and information structure entries from the two or more of the class files without duplication of entries* (see at least Paragraph 0084, "The methods and fields from the original class fields are copied to the bundle as well along with various other class file objects (not shown)." Applicant regards "byte codes and information structure" as the "class properties, the methods, fields and attributes of the class, and their types" (Paragraph 0013). The constant pool contains references to these byte codes and information structures. Therefore, if the constant pool entities have been combined into the bundle without duplication, the methods, fields, and attributes are copied into the bundle without duplication as well.).

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention was made to incorporate Swetland's, memory and computing units and process of merging the constant pools and class structures of related methods, into Baentsch's, fixup table. The modification would be obvious because one of ordinary skill in the art would be motivated to "[reduce] the memory requirements for object-oriented programs" (Swetland, Paragraph 0011), "considering the fact that a program may utilize scores of class files" (Swetland, Paragraph 0010). By merging the constant pools and class structures of related methods, the memory requirements for programs will be reduced since there can be an enormous number of class files, and hence methods, in a program. Baentsch's fixup table allows for "runtime efficient linking in resource constraint Java runtime environments" (Baentsch, Page 1: 26-27). Applicant's cod file is essentially a compressed class file, and Baentsch's cap file is a package file that contains classes and interfaces. Both files contain class information and for the purposes of combining class information and one of ordinary skill in the art would view these as comparable file types.

As per **Claim 14**, the rejection of **Claim 13** is incorporated; and Baentsch further discloses:

- *wherein for the given cod file, the fixup table includes a symbolic reference for cross-referencing a method not contained in the cod file* (see at least Page 4, lines 22-24, "…references to other external packages should not be linked by precalculated offsets.

Instead, a name or identifier should be used for references to other packages during the link process.").

As per **Claim 15**, the rejection of **Claim 13** is incorporated; and Baentsch further discloses:

- *and the fixup table of the given cod file further includes indices to the other related cod files specified in the sibling group* (see at least Page 4, lines 12-14, "The fixup table...contains the position in the text or data section where a relocation has to take place. In the simple case, these places are also relocated by a precalculated offset into trusted and well known target packages.").

Baentsch does not disclose:

- *wherein the given cod file further includes a sibling list for listing other related cod files to define a sibling group*

Swetland discloses:

- *wherein the given cod file further includes a sibling list for listing other related cod files to define a sibling group* (see at least Figure 12; Paragraph 0083, "…each of the class files used in a particular application program…are combined to form a unified programming object referred to herein as a 'bundle'. For the purpose of illustration, the particular bundle...is constructed from the class files." Related class files are combined to form a bundle, which one of ordinary skill in the art would view as comparable to a sibling group.)

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention was made to incorporate Swetland's bundled class files into Baentsch's fixup table containing references to the related files. The modification would be obvious because one of ordinary skill in the art would be motivated to group related class files to form a unified class structure in a medium that limits the size of an individual file by having multiple smaller related files as opposed to a single larger combined file.

As per **Claim 16**, the rejection of **Claim 13** is incorporated; and Baentsch further discloses:

- *wherein for the given cod file, the executable software comprises code for generating a hard offset in the byte codes and information structure for cross-referencing a method included in the given cod file that was previously symbolically referenced* (see at least Page 4, lines 12-14, "The fixup table...contains the position in the text or data section where a relocation has to take place. In the simple case, these places are also relocated by a precalculated offset into trusted and well known target packages.").

As per **Claim 17**, the rejection of **Claim 15** is incorporated; and Baentsch further discloses:

- *wherein for the given cod file, the executable software comprises code for generating a hard offset in the byte codes and information structure for cross-referencing a method included in the sibling group that was previously symbolically referenced* (see at least Page 4, lines 12-14, "The fixup table...contains the position in the text or data section where a relocation has to take place. In the simple case, these places are also relocated by a precalculated offset into trusted and well known target packages." It would have been obvious to one of ordinary skill in the art to be able to use hard-offsets in a file to reference other related files, or sibling files.).

As per **Claim 18**, the rejection of **Claim 15** is incorporated; and Baentsch further discloses:

- *wherein for the given cod file, the executable software comprises code for generating a symbolic reference in the fixup table for cross-referencing a method not contained in the sibling group* (see at least Page 4, lines 22-24, "…references to other external packages should not be linked by precalculated offsets. Instead, a name or identifier should be used for references to other packages during the link process." These references made to external packages can include methods.).

Conclusion

18. Any inquiry of a general nature or relating to the status of this application or concerning this communication or earlier communications from the examiner should be directed to Kimberly Jordan whose telephone number is 571-270-5481. The examiner can normally be reached on Monday-Friday 9:30am-5pm EST. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, James Reagan can be reached on 571-272-6710.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

November 21, 2008

/Kimberly Jordan/

Examiner, Art Unit 4114

/James A. Reagan/

Supervisory Patent Examiner, Art Unit 4114